



MODUL PRAKTIKUM

PEMROGRAMAN BERBASIS FRAMEWORK

Modul 12

- Operasi CRUD menggunakan *Firebase Realtime Database*

JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

Operasi CRUD API pada *Firebase Database Realtime*

Pada modul sebelumnya, kita sudah mempelajari login menggunakan firebase. Untuk materi kali ini kita akan mempelajari proses CRUD menggunakan *firebase*. Terutama operasi *Create*, *Read*, dan *Delete*. Dalam modul ini kita akan membuat perbandingan antara *Global API service* dengan *Fake API server* (Modul 8 – Global API) dengan API service dari *firebase realtime database*.

Sebelum memulai praktikum, silahkan persiapkan source program yang telah kalian kerjakan pada praktikum Modul-8 (Global API). Atau bisa juga download di LMS untuk file [Source Modul 8.rar](#).

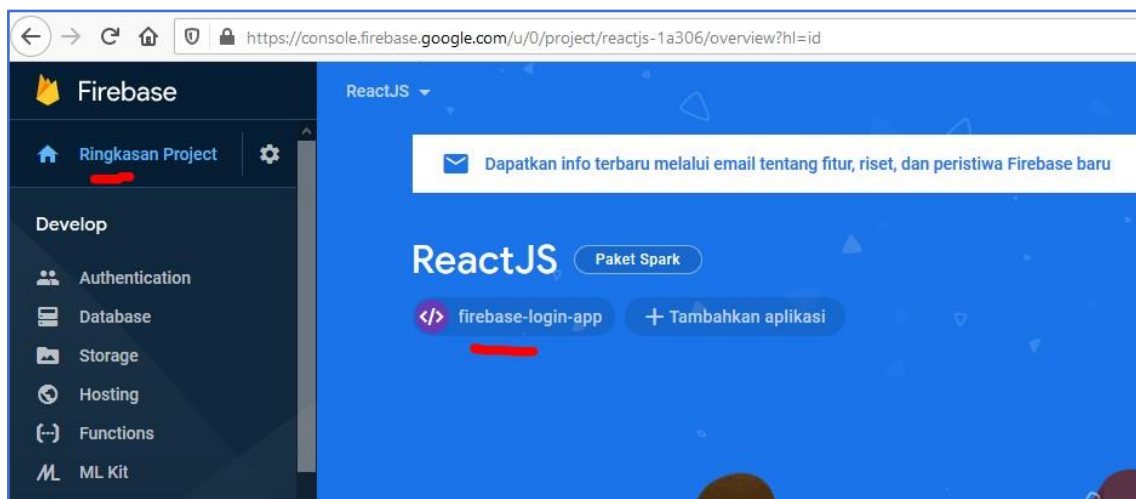
Praktikum

CRUD pada Firebase

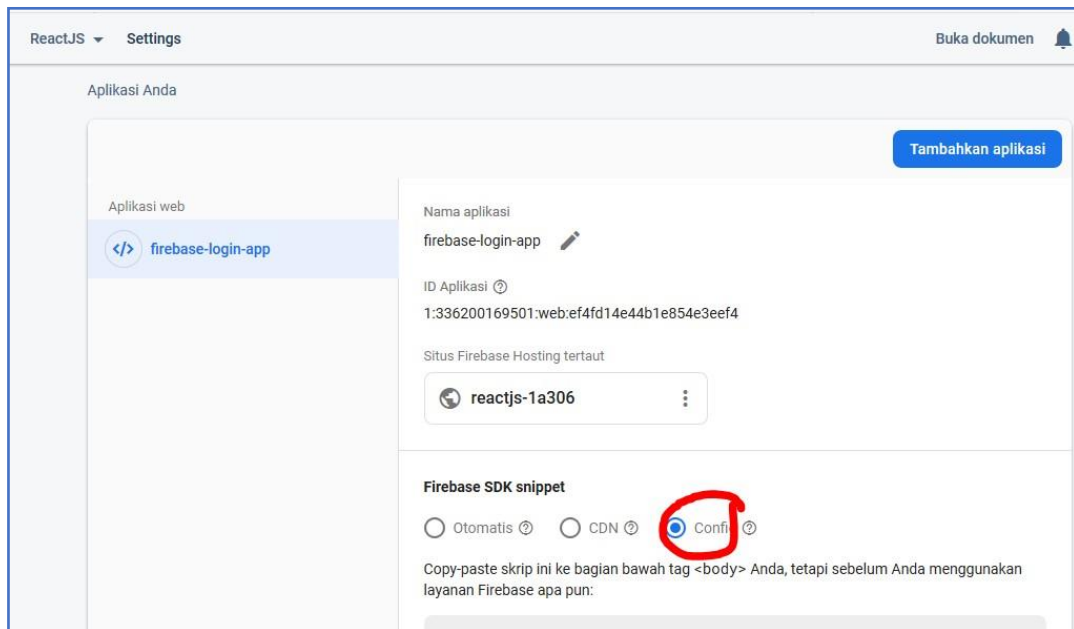
1.1 Langkah Persiapan

Dalam pembuatan Create method pada Firebase, maka perlu database yang akan digunakan untuk menyimpan hasil insert data yang kita masukkan. Langkah-langkah pada persiapan praktikum ini adalah

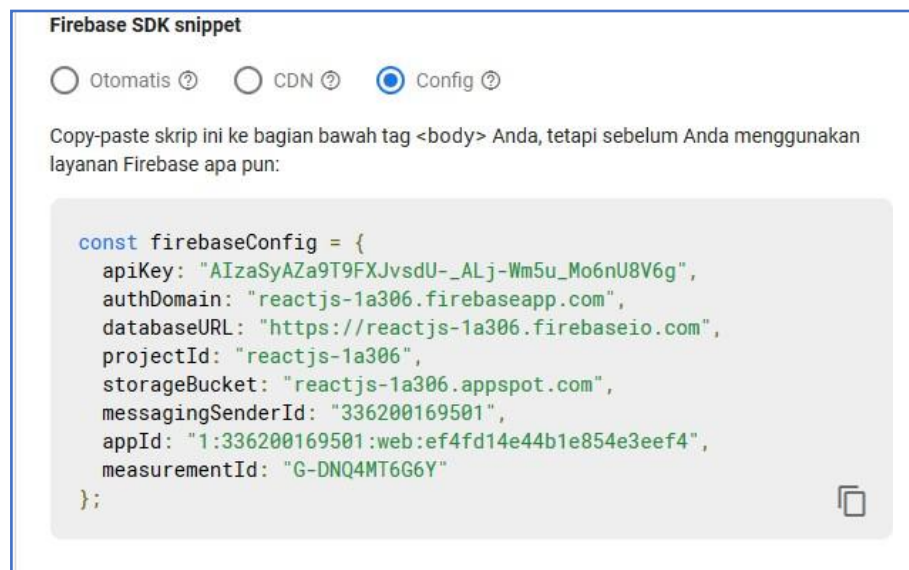
1. Buat project **reactjs** baru, atau pakai project **reactjs** yang sudah ada.
2. Backup folder **public** dan **src**, kemudian hapus semua file dan folder yang ada di dalam folder **public** dan **src**.
3. Ekstrak file **Sorce Modul 8.rar** ke dalam project **reactjs**.
4. Silahkan install beberapa *package* yang dibutuhkan melalui CMD, seperti
 - a. **npm install react-router-dom**
 - b. **npm install --save firebase**
 - c. **npm install bootstrap**
5. Pastikan anda tidak lupa konfigurasi dari aplikasi firebase yang telah dibuat. Jika lupa bisa masuk project firebase konsol di <https://console.firebase.google.com>
6. Kemudian klik **"ringkasan project"**, dan pilih app yang sudah dibuat. Sesuai dengan materi sebelumnya adalah app **"firebase-login-app"**, kemudian klik setting.



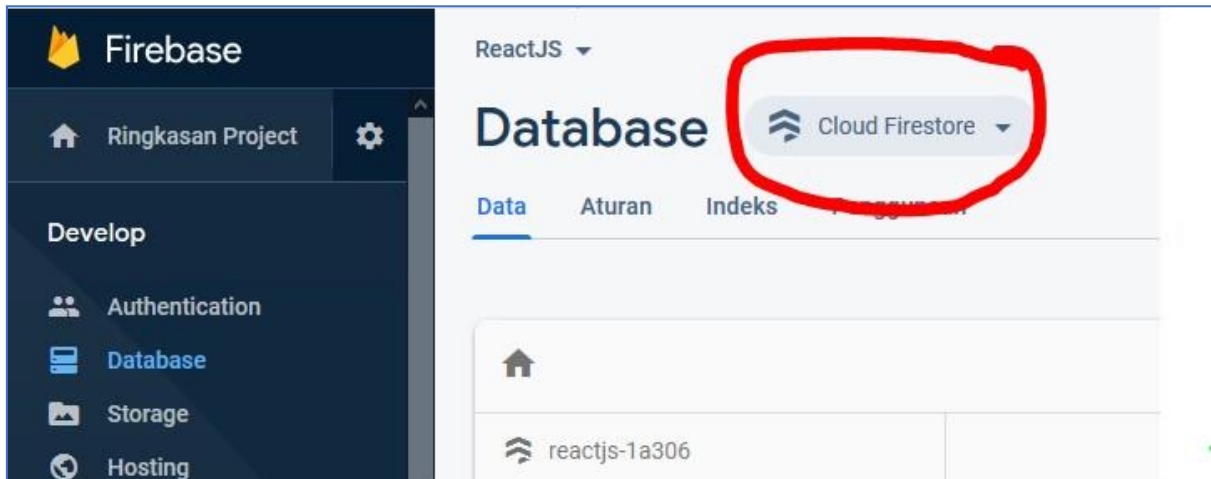
7. Pada tab **umum/common** di menu **Setting**, scroll ke bawah, dan pilih opsi **config** pada **Firestore SDK snippet**.



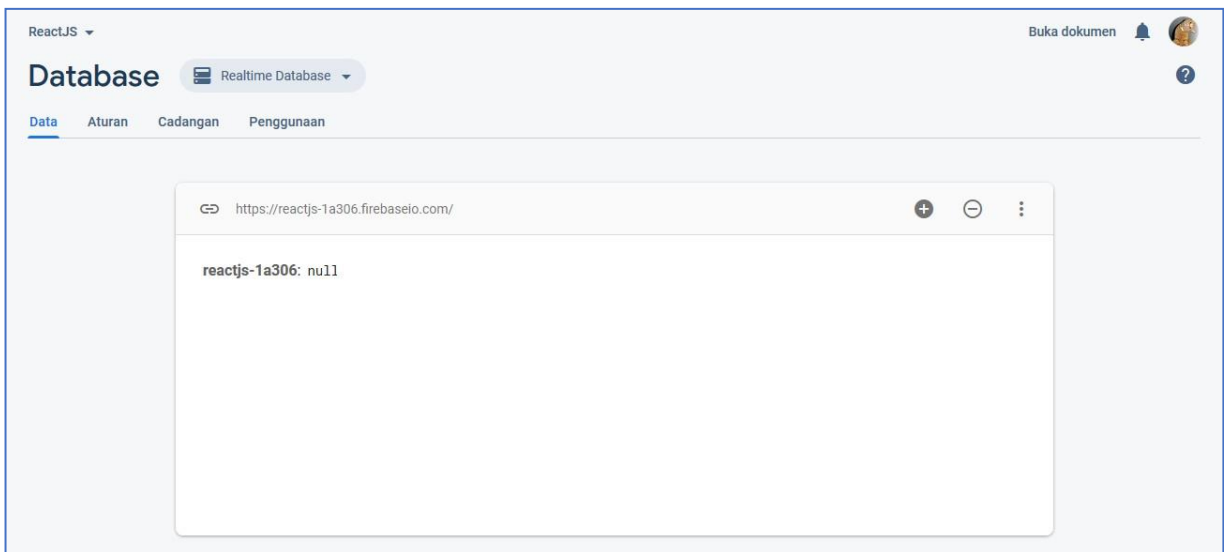
8. Konfigurasi inilah yang akan terus kita gunakan saat berinteraksi dengan API Firebase yang telah kita buat.



9. Masuk pada menu **database** pada firebase. Pada materi kali ini kita menggunakan **real time database firebase** jadi pilih menu **Real time database**.



10. Seperti Gambar berikut tampilan database real time firebase yang akan kita gunakan

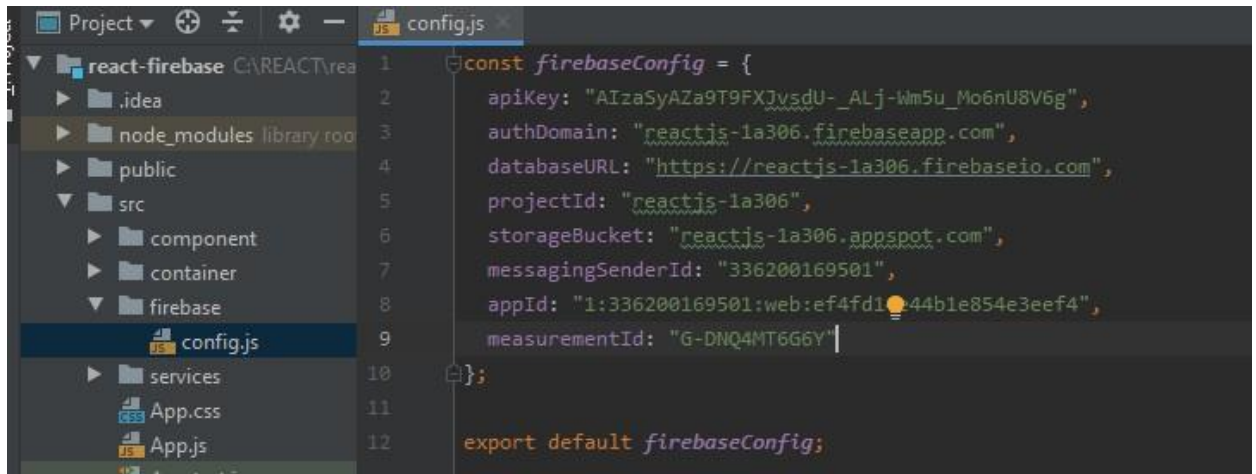


11. Langkah Persiapan telah selesai. Kita mulai praktikumnya.

1.2 Langkah Praktikum

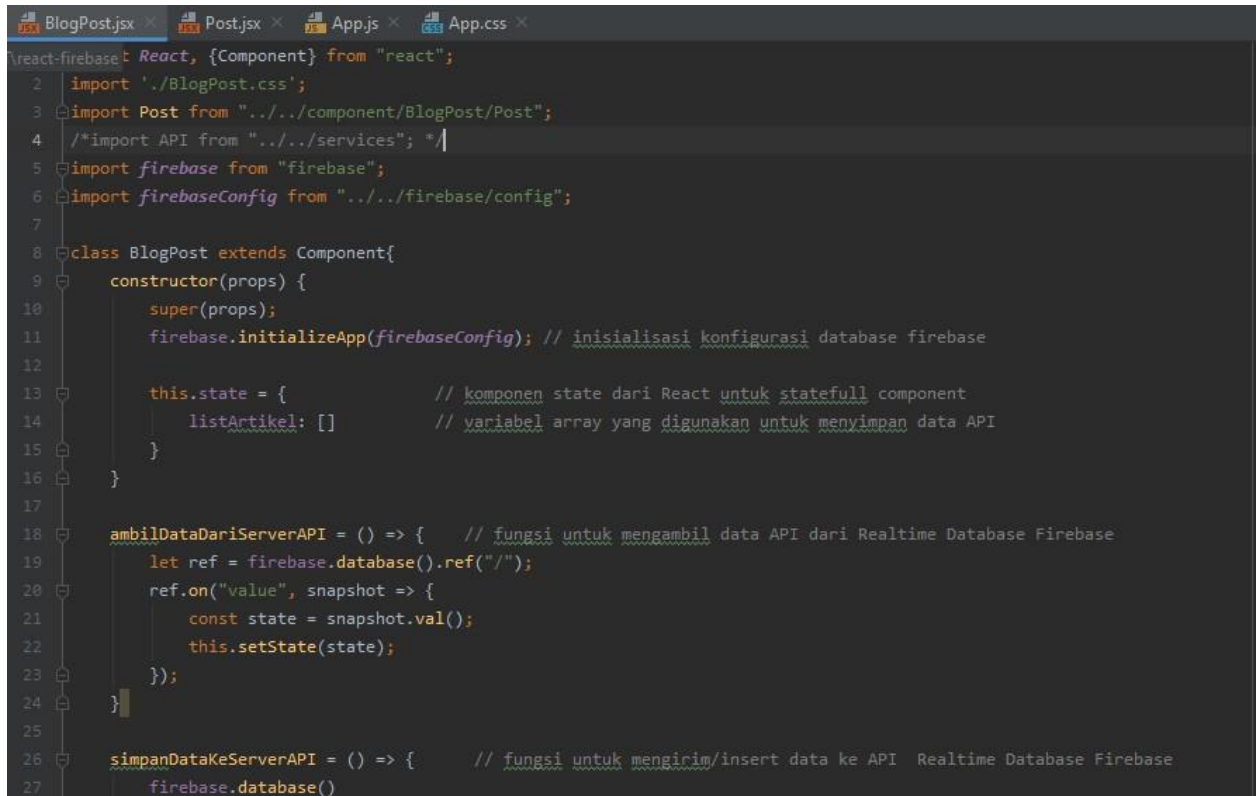
Praktikum ini sedikit membahas antara praktikum [Modul 8 \(Global API\)](#) dengan [Modul 10 dan 11](#) tentang [Firebase](#). Langkah-langkah praktikum yang dikerjakan adalah

12. Pada project ReactJS kalian, buat file dan folder `firebase/config.js` dalam folder `src` dan simpan konfigurasi code program di atas (seperti Langkah 8) pada file `config.js`.



```
1 const firebaseConfig = {
2   apiKey: "AIzaSyAZa9T9FXJv5dU-_ALj-Wm5u_Mo6nU8V6g",
3   authDomain: "reactjs-1a306.firebaseio.com",
4   databaseURL: "https://reactjs-1a306.firebaseio.com",
5   projectId: "reactjs-1a306",
6   storageBucket: "reactjs-1a306.appspot.com",
7   messagingSenderId: "336200169501",
8   appId: "1:336200169501:web:ef4fd1444b1e854e3eef4",
9   measurementId: "G-DNQ4MT6G6Y"
10 };
11
12 export default firebaseConfig;
```

13. Kita modifikasi *statefull component* `BlogPost.jsx` yang dulu pernah kita kerjakan, menjadi seperti ini



```
1 import React, {Component} from "react";
2 import './BlogPost.css';
3 import Post from "../../component/BlogPost/Post";
4 /*import API from "../../services"; */
5 import firebase from "firebase";
6 import firebaseConfig from "../../firebase/config";
7
8 class BlogPost extends Component{
9   constructor(props) {
10     super(props);
11     firebase.initializeApp(firebaseConfig); // inisialisasi konfigurasi database firebase
12
13     this.state = { // komponen state dari React untuk statefull component
14       listArtikel: [] // variabel array yang digunakan untuk menyimpan data API
15     }
16   }
17
18   ambilDataDariServerAPI = () => { // fungsi untuk mengambil data API dari Realtime Database Firebase
19     let ref = firebase.database().ref("/");
20     ref.on("value", snapshot => {
21       const state = snapshot.val();
22       this.setState(state);
23     });
24   }
25
26   simpanDataKeServerAPI = () => { // fungsi untuk mengirim/insert data ke API Realtime Database Firebase
27     firebase.database()
```

```

BlogPost.jsx
26  simpanDataKeServerAPI = () => {      // fungsi untuk mengirim/insert data ke API Realtime Database Firebase
27      firebase.database()
28      .ref("/")
29      .set(this.state);
30  }
31
32  componentDidMount() {      // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API
33      this.ambilDataDariServerAPI() // ambil data dari server API lokal
34  }
35
36  componentDidUpdate(prevProps, prevState) {
37      if (prevState !== this.state) {
38          this.simpanDataKeServerAPI();
39      }
40  }
41
42  handleHapusArtikel = (idArtikel) => {      // fungsi yang meng-handle button action hapus data
43      const {listArtikel} = this.state;
44      const newState = listArtikel.filter(data => {
45          return data.uid !== idArtikel;
46      });
47      this.setState({listArtikel: newState});
48  }
49
50  handleTombolSimpan = (event) => {      // fungsi untuk meng-handle saat tombol submit d klik
51      let title = this.refs.judulArtikel.value; // this.refs mengacu pada input field (input text, textarea, dll)
52      let body = this.refs.isiArtikel.value;
53      let uid = this.refs.uid.value;

```

```

BlogPost.jsx
50  handleTombolSimpan = (event) => {      // fungsi untuk meng-handle saat tombol submit d klik
51      let title = this.refs.judulArtikel.value; // this.refs mengacu pada input field (input text, textarea, dll)
52      let body = this.refs.isiArtikel.value;
53      let uid = this.refs.uid.value;
54
55      if (uid && title && body) { // Cek apakah semua data memiliki nilai (tidak null)
56          const { listArtikel } = this.state;
57          const indeksArtikel = listArtikel.findIndex(data => {
58              return data.uid === uid;
59          });
60          listArtikel[indeksArtikel].title = title;
61          listArtikel[indeksArtikel].body = body;
62          this.setState({ listArtikel });
63      } else if (title && body) { // jika data belum ada di server
64          const uid = new Date().getTime().toString();
65          const { listArtikel } = this.state;
66          listArtikel.push({ uid, title, body });
67          this.setState({ listArtikel });
68      }
69
70      this.refs.judulArtikel.value = "";
71      this.refs.isiArtikel.value = "";
72      this.refs.uid.value = "";
73  };
74
75  render() {
76      return(
77          <div className="post-artikel">

```



```

BlogPost.jsx
render() {
  76   return(
  77     <div className="post-artikel">
  78       <div className="form pb-2 border-bottom">
  79         <div className="form-group row">
  80           <label htmlFor="title" className="col-sm-2 col-form-label">Judul</label>
  81           <div className="col-sm-10">
  82             <input type="text" className="form-control" id="title" name="title" ref="judulArtikel"/>
  83           </div>
  84         </div>
  85         <div className="form-group row">
  86           <label htmlFor="body" className="col-sm-2 col-form-label">Isi</label>
  87           <div className="col-sm-10">
  88             <textarea className="form-control" id="body" name="body" rows="3" ref="isiArtikel"></textarea>
  89           </div>
  90         </div>
  91         <input type="hidden" name="uid" ref="uid" />
  92         <button type="submit" className="btn btn-primary" onClick={this.handleTombolSimpan}>Simpan</button>
  93       </div>
  94       <h2>Daftar Artikel</h2>
  95       { // looping dan masukkan untuk setiap data yang ada di listArtikel ke variabel artikel
  96         this.state.listArtikel.map(artikel => {
  97           return <Post key={artikel.uid} judul={artikel.title} isi={artikel.body}
  98             idArtikel={artikel.uid} hapusArtikel={this.handleHapusArtikel}/>
  99         })
  100       }
  101     </div>
  102   )
  103 }
  104
  105
  106 export default BlogPost;

```

14. Edit pula *stateless component* `Post.jsx` menjadi seperti gambar berikut


```
BlogPost.jsx Post.jsx
1 import React from "react";
2
3 const Post = (props) => {
4   return (
5     <div className="artikel">
6       <div className="gambar-artikel">
7         
8       </div>
9       <div className="konten-artikel">
10        <div className="judul-artikel">{props.judul}</div>
11        <p className="isi-artikel">{props.isi}</p>
12        <button className="btn btn-sm btn-warning"
13          onClick={() => { if (window.confirm('Apakah anda yakin menghapus artikel ini?')) props.hapusArtikel(props.idArtikel)}} >
14          Hapus
15        </button>
16      </div>
17    </div>
18  )
19 }
20
21 export default Post;
```

15. Silahkan lakukan proses insert data pada browser, lihat apa yang terjadi

1.3 Pertanyaan Praktikum

- Perhatikan file **BlogPost.jsx**, apa saja kode program yang berubah dari **BlogPost.js** pada Modul-8 dengan **BlogPost.jsx** pada praktikum kali ini? Kenapa?
- Perhatikan file **Post.jsx**, apa saja kode program yang berubah dari **Post.js** pada Modul-8 dengan **Post.jsx** pada praktikum kali ini? Kenapa?
- Apakah **Global API service** yang kita buat pada Modul-8 kemarin kita pakai lagi pada praktikum kali ini? Kenapa alasannya?
- Data yang kita insert bertambah, dan saat kita refresh browser, data masih tetap ada. Dimanakah data-data artikel tersebut disimpan? Tunjukkan hasil screenshot data disimpan tersebut.
- Menurut kalian lebih mudah dan lebih praktis mana aplikasi reactjs menggunakan data API sendiri (seperti Modul-4 dan Modul-8) atau menggunakan Firebase realtime database? Berika alasannya.

~ ~ Selamat Mengerjakan ~ ~ ~